


© 2004-2009 Volnys Bernal 1

Condições de Disputa

Volnys Borges Bernal
volnys@lsi.usp.br
<http://www.lsi.usp.br/~volnys>



© 2004-2009 Volnys Bernal 2

Agenda

- ❑ Condições de disputa
- ❑ Região Crítica


© 2004-2009 Volnys Bernal 3

Sobre esta apresentação

- ❑ Esta apresentação ...
 - ❖ Não apresenta todos os detalhes sobre este tópico.
 - ❖ É um resumo para auxiliar a apresentação do tópico em sala de aula.
- ❑ Para estudo, deve ser utilizada uma das seguintes referências:
 - ❖ Capítulos 1 e 2 do livro:
 - ANDREW S. TANENBAUM; Sistemas Operacionais Modernos. Prentice-Hall
 - ❖ Capítulos 1 e 2 do livro:
 - ANDREW S. TANENBAUM; Sistemas Operacionais. Prentice-Hall.

© 2004-2009 Volnys Bernal 4

Condições de disputa (Race Conditions)



© 2004-2009 Volnys Bernal 5

Condições de disputa

- ❑ Condição de disputa é
 - ❖ Uma situação de conflito ...
 - ❖ No acesso a um determinado recurso (variável, arquivo, ...)
 - ❖ Recurso este compartilhado
 - ❖ Por duas ou mais entidades de processamento (processos, *threads*, ...)
 - ❖ Que pode causar situações não desejáveis e resultados não esperados
- ❑ Importante:
 - ❖ Threads de um mesmo processo possuem diversos recursos compartilhados
 - Área de dados
 - Arquivos abertos
 - etc
 - ❖ Quando existem acessos de escrita a estes recursos compartilhados podem ocorrer potenciais situações de condição de disputa


© 2004-2009 Volnys Bernal 6

Condições de disputa

- ❑ Existem inúmeras situações na qual existe condição de disputa.
- ❑ A seguir, serão apresentados 3 exemplos de condição de disputa:
 - ❖ Exemplo 1: Contador
 - ❖ Exemplo 2: Manipulação de lista ligada
 - ❖ Exemplo 3: Variável de proteção

© 2004-2009 Volnys Bernal 7

Exemplo 1: Contador



© 2004-2009 Volnys Bernal 8

Exemplo 1: Contador

❑ **Descrição**

- ❖ Dois *threads* realizam determinadas tarefas.
- ❖ Após realizar cada tarefa incrementam um contador *c*.
- ❖ Variável *c* é global (compartilhada entre os dois threads)

Thread1:

```
...
while (1)
  <Realiza tarefa>
  c = c + 1
...
```

Thread2:

```
...
while (1)
  <Realiza tarefa>
  c = c + 1
...
```

© 2004-2009 Volnys Bernal 9

Exemplo 1: Contador

❑ **Versão do programa em assembler**

Thread1:

```
...
repete: ...
realiza tarefa
...
LOAD AC, (c)
ADD AC, 1
STORE (c), AC
JUMP repete
...
```

Thread2:

```
...
repete: ...
realiza tarefa
...
LOAD AC, (c)
ADD AC, 1
STORE (c), AC
JUMP repete
...
```

© 2004-2009 Volnys Bernal 10

Exemplo 1: Contador

❑ **Problemas: condição de disputa sobre o contador "c"**

- ❖ **Sistemas monoprocessoadores**
 - Concorrência:
 - troca de contexto durante a atualização do contador "c"
- ❖ **Sistemas multiprocessadores**
 - Concorrência:
 - troca de contexto durante a atualização do contador "c"
 - Paralelismo:
 - incremento simultâneo do contador "c"

© 2004-2009 Volnys Bernal 11

Exemplo 1: Contador

❑ **Condição de disputa na concorrência:**

Thread1:

```
...
repete: ...
realiza tarefa
...
LOAD AC, (c)
ADD AC, 1
STORE (c), AC
JUMP repete
...
```

1
↓
3

Thread2:

```
...
repete: ...
realiza tarefa
...
LOAD AC, (c)
ADD AC, 1
STORE (c), AC
JUMP repete
...
```

2
↓

© 2004-2009 Volnys Bernal 12

Exemplo 1: Contador

❑ **Condição de disputa no paralelismo**

Thread1:


```
...
repete: ...
realiza tarefa
...
LOAD ↓1 AC, (c)
ADD ↓2 AC, 1
STORE ↓3 (c), AC
JUMP repete
...
```

Thread2:

```
...
repete: ...
realiza tarefa
...
LOAD ↓1 AC, (c)
ADD ↓2 AC, 1
STORE ↓3 (c), AC
JUMP repete
...
```

© 2004-2009 Volnys Bernal 13

Exemplo 2: Lista ligada



© 2004-2009 Volnys Bernal 14

Exemplo 2: Lista ligada

- Exemplo 2: Manipulação de lista ligada
 - ❖ Dois threads manipulam (leitura ou modificação) uma lista ligada

Thread1:

```
...
<manipula lista ligada>
...
```

Thread2:

```
...
<manipula lista ligada>
...
```


© 2004-2009 Volnys Bernal 15

Exemplo 2: Lista ligada

- Problema: condição de disputa durante a manipulação da lista ligada:
 - ❖ Sistemas monoprocessoadores
 - Concorrência: Troca de contexto durante a atualização da lista deixa-a em um estado inconsistente
 - ❖ Sistemas multiprocessoadores
 - Concorrência:
 - Troca de contexto durante a atualização da lista deixa-a em um estado inconsistente
 - Paralelismo:
 - Dois threads utilizando a lista simultaneamente e, pelo menos 1 deles alterando-a

© 2004-2009 Volnys Bernal 16

Exemplo 3: Variável de proteção



© 2004-2009 Volnys Bernal 17

Exemplo 3: Variável de proteção

- Descrição
 - ❖ Dois threads definem uma variável compartilhada para controle do uso do recurso.
 - ❖ Se a variável for 1 significa que o recurso está ocupado, se for zero está livre.

Thread1:

```
...
while (ocupado == 1);
ocupado = 1;
<usa recurso>
ocupado = 0;
...
```

Thread2:

```
...
while (ocupado == 1);
ocupado = 1;
<usa recurso>
ocupado = 0;
...
```

© 2004-2009 Volnys Bernal 18

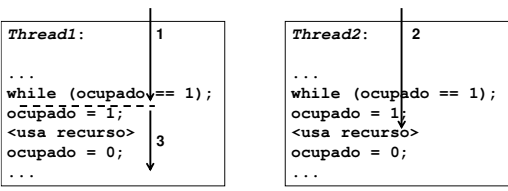
Exemplo 3: Variável de proteção

Thread1:

```
...
while (ocupado == 1);
ocupado = 1;
<usa recurso>
ocupado = 0;
...
```

Thread2:

```
...
while (ocupado == 1);
ocupado = 1;
<usa recurso>
ocupado = 0;
...
```



© 2004-2009 Volnys Bernal 19

Exemplo 3: Variável de proteção

- ❑ **Problema:**
 - ❖ Condição de disputa sobre a variável "ocupado"
 - ❖ **Sistemas monoprocessadores**
 - Concorrência:
 - Troca de contexto durante a alteração da variável "ocupado" para 1
 - ❖ **Sistemas multiprocessadores**
 - Concorrência:
 - Troca de contexto durante a alteração da variável "ocupado" para 1
 - Paralelismo:
 - Dois *threads* alterando simultaneamente a variável "ocupado" para 1

© 2004-2009 Volnys Bernal 20

Exemplo 3: Variável de proteção

- ❑ **Condição de disputa quando existe concorrência**

Thread1:

```

...
while (ocupado == 1);
ocupado = 1;
<usa recurso>
ocupado = 0;
...
                
```

↓ 1

↓ 3

Thread2:

```

...
while (ocupado == 1);
ocupado = 1;
<usa recurso>
ocupado = 0;
...
                
```

↓ 2

↓ 4

© 2004-2009 Volnys Bernal 21

Exemplo 3: Variável de proteção

- ❑ **Condição de disputa quando existe paralelismo**

Thread1:

```

...
while (ocupado == 1);
ocupado = 1;
<usa recurso>
ocupado = 0;
...
                
```

↓ 1

↓ 2

Thread2:

```


...
while (ocupado == 1);
ocupado = 1;
<usa recurso>
ocupado = 0;
...
                
```

↓ 1

↓ 2

© 2004-2009 Volnys Bernal 22

Região Crítica



© 2004-2009 Volnys Bernal 23

Região Crítica

- ❑ **Região crítica é ...**
 - ❖ Uma região de código ...
 - ❖ Na qual existe acesso a recursos compartilhados ...
 - ❖ Na qual existe condição de disputa
- ❑ **Objetivo da região crítica**
 - ❖ Identificar a região de código na qual existe potencialmente ocorrência de condição de disputa devido ao acesso por duas ou mais entidades
 - ❖ Possibilitar a utilização de soluções de sincronização para evitar condição de disputa na região crítica
- ❑ **Obs:**
 - ❖ Entidades
 - Processos, threads, ...

© 2004-2009 Volnys Bernal 24

Região Crítica

- ❑ **Exemplo de região de código na qual existe acesso a recursos compartilhados que pode causar problema de condição de disputa**

```

...
<manipula recurso compartilhado>
...
                
```

```

...
<manipula recurso compartilhado>
...
                
```

↓
Região Crítica

© 2004-2009 Volnys Bernal 25

Região Crítica

❑ Exemplo 1:
❖ Contador de tarefas

Thread1:

```
...
while (1)
  <Realiza tarefa>
  c = c + 1
...
```

Thread2:

```
...
while (1)
  <Realiza tarefa>
  c = c + 1
...
```

RC_1

© 2004-2009 Volnys Bernal 26

Região Crítica

❑ Exemplo 2:
❖ Manipulação de lista ligada

Thread1:

```
...
<manipula lista ligada>
...
```

Thread2:

```
...
<manipula lista ligada>
...
```

RC_1

© 2004-2009 Volnys Bernal 27

Região Crítica

❑ Exemplo 3:
❖ Variável de proteção

Thread1:

```
...
while (ocupado == 1);
ocupado = 1;
<usa recurso>
ocupado = 0;
...
```


Thread2:

```
...
while (ocupado == 1);
ocupado = 1;
<usa recurso>
ocupado = 0;
...
```

RC_1

© 2004-2009 Volnys Bernal 28

Referências Bibliográficas



© 2004-2009 Volnys Bernal 29

Referências Bibliográficas

- ❑ ANDREW S. TANENBAUM; **Sistemas Operacionais Modernos**. Prentice-Hall.
❖ Capítulo 2
- ❑ ANDREW S. TANENBAUM; **Sistemas Operacionais**. Prentice-Hall.
❖ Capítulo 2